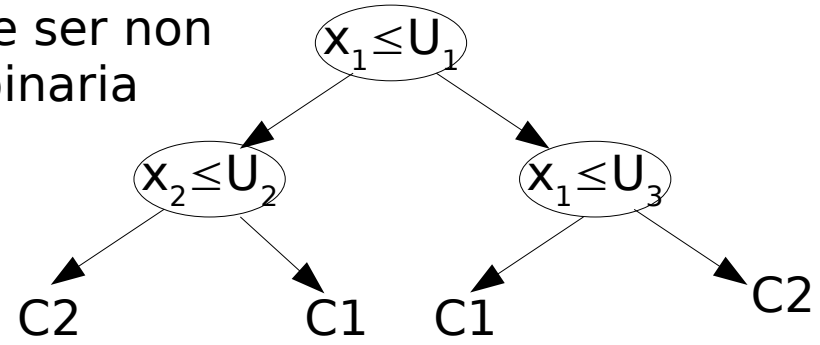
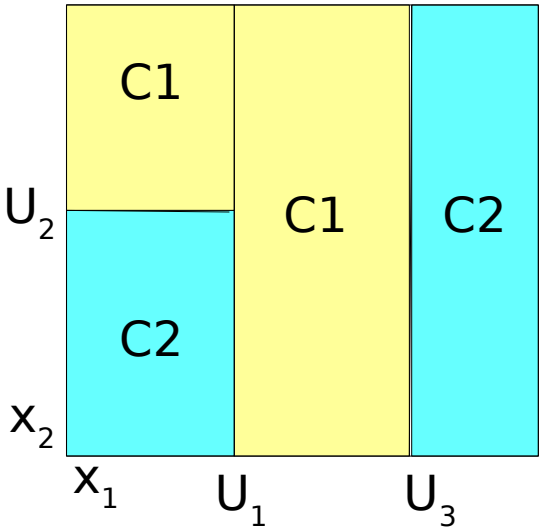
Tema 3. Árbores de clasificación e regresión

Árbore binaria: pode ser non binaria





- Cada nodo divide (splits) unha entrada x_i de $\mathbf{x}=(x_1,\ldots,x_n)$ usando un umbral U_i. Primeiro selecciona a entrada i
- Representación como árbore (esquerda) ou como rexións no espazo de entrada (dereita)
- Fronteiras entre rexións paralelas aos eixos, definidas polos umbrais U_i

Creación de árbore de clasificación binaria

proc árbore(r,$\{\mathbf{x}_{s(i)},y_{s(i)}\}_{i=1}^{M}$) # r=nodo raíz, $\{\mathbf{x}_{s(i)},y_{s(i)}\}_{i=1}^{M}$=patróns de r
  if ∃y tal que y_i=y para i=1..M # tódolos $\mathbf{x}_i$ da mesma clase
    y_r=y; r terminal; **remata**
  else
    avalía incr. entropía nodo r : $E=-\sum_{i=1}^{M} P_i \log_2 P_i$;ΔE=0; k=1; U=1
    for i=1:n # percorre entradas
      for j=$\mathbf{v}_i$ #$\mathbf{v}_i$=vector cos valores de característica i
        avalía entropía E_{ij} dividindo a entrada x_i co umbral j
        if E-E_{ij}>ΔE; ΔE=E-E_{ij}; k=i; U=j #k=entrada;U=umbral
      end
    end
    δ(t)=1 se t=0 δ(t)=0 noutro caso
    if ΔE=0
      $y_r = \underset{j=1\ldots C}{argmax} \left\{ \sum_{i=1}^{M} \delta(j - y_{s(i)}) \right\}$; r terminal; **remata**
    else
      crea nodo r_e; árbore(r_e,$\{\mathbf{x}_{ij},y_i : x_{ik}<U\}$) # recursivo nodo esquerdo
      crea nodo r_d; árbore(r_d,$\{\mathbf{x}_i,y_i : x_{ik}\geq U\}$) # recursivo nodo dereito
    endif
  end if
end proc

Asocia o nodo terminal á clase máis poboada

Entrenamento recursivo dende arriba (nodo raíz) cara abaixo na árbore

Saída da árbore de clasificación

- Sexa R o nº de rexións $R_1..R_R$ no espazo de entrada; $I_i(\mathbf{x})=1$ se $\mathbf{x}\in R_i$ e $I_i(\mathbf{x})=0$ noutro caso; R_i a rexión i-ésima, con i=1..R
- A saída da árbore é: $z(\boldsymbol{x})=\sum_{i=1}^{R} y_i I_i(\boldsymbol{x})$
- Na práctica, o procedemento que se segue é:

1) O patrón de teste **x** preséntase ao nodo raíz r, que divide a entrada k con umbral U e efectúase o teste $x_k>U$

2) Se $x_k<U$, repítese o proceso co nodo r_e (fillo esquerdo)

3) Se $x_k\geq U$, repítese o proceso co nodo r_d (fillo dereito)

4) O proceso repítese ata que se acada un nodo r terminal

5) A saída da árbore é a etiqueta y_r que durante o entrenamento foi a clase maioritaria entre os patróns deste nodo terminal r

Selección da entrada a dividir

- **Entropía** E: avalía en que medida os patróns de entrenamento que chegan a un nodo teñen distintas clases
- Sexan: N_i=nº patróns de clase i no nodo; N=nº patróns do nodo; $P_i=N_i/N$, a proporción de patróns da clase i no nodo:

$$E=-\sum_{i=1}^{C} P_i \log_2 P_i$$

- A mellor entrada j e umbral U: as que máis reducen a entropía E:

$$(j,U)=\underset{1\le i\le n;\, k\in \boldsymbol{v}_i}{argmax}\{\Delta E_{ik}\} \qquad \Delta E_{ik}=E-P\,E_{ik}^{e}-(1-P)\,E_{ik}^{d}$$

- $\mathbf{v}_i$=valores da entrada x_i para os patróns do nodo
- $P=N_e/N$, $N_e(i,k)$=nº de patróns asignados ao nodo esquerdo (e)
- E_{ik}^{e}, E_{ik}^{d}: entropías dos nodos esquerdo e dereito (entrada x_i e umbral k)

Medidas de erro alternativas á entropía

- **Índice de Gini**: $G = 1 - \sum_{i=1}^{C} P_i^2$: erro medio se seleccionas a clase aleatoriamente entre as presentes no nodo
- **Erro de clasificación**: $J = 1 - \max_{1 \le i \le C} \{P_i\}$

  J=0 so se ∃m con P_m=1, i.e., se N_m=1 e N_i=0 para i≠m (tódolos patróns que chegan ao nodo son da mesma clase)
- Os criterios de parada e podado inflúen máis nos resultados que medida de erro usada
- Unha **entrada continua** discretízase en S=T+1 intervalos de lonxitudes iguais ou distintas usando unha serie de umbrais $\{t_i\}_{i=1}^{T}$. Cada nodo divide en S nodos
- Os umbrais t_i fíxanse: 1) equiespazados no rango de valores; ou 2) maximizando a varianza de cada intervalo neste rango

Criterios de parada na creación de nodos

1) O erro de validación deixa de reducirse
2) A mellor división ten $\Delta E < \Delta E_0$
3) Non se engaden máis nodos
4) Ao acadar un nº máximo de nodos

$$\Delta E = \frac{E - \sum_{i=1}^{B} P_i E_i}{-\sum_{i=1}^{B} P_i \log_2 P_i}$$

- As árbores tenden á sobreaprendizaxe
- A árbore pode non ser binaria usando un factor de división B>2
- Selecciónase os valores (i,U) que maximizan este incremento da entropía.
- P_r=fracción de patróns do nodo asignados ao fillo r
- Obxectivo: árbore simple e compacta con poucos nodos

Regularización e podado

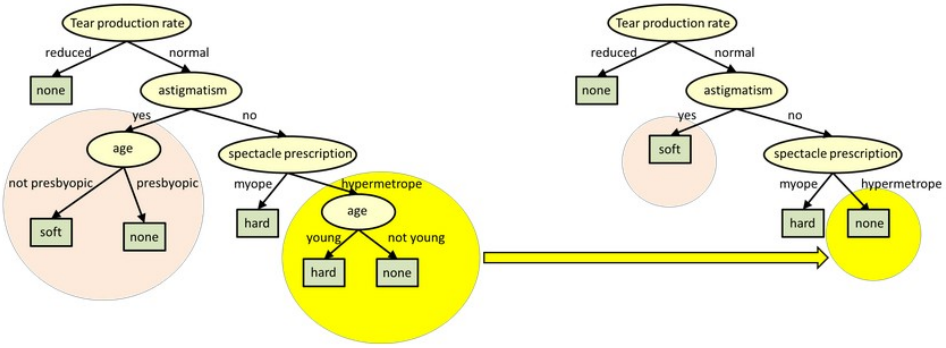
- **Regularización**: consiste en ponderar a suma das entropías dos M nodos terminais e o tamano T da árbore (nº nodos ou enlaces)

$$\sum_{i=1}^{M} E_i + \lambda T$$

entropía dos M nodos terminais sobre o conxunto de entrenamento

tamano da árbore

- **Podado** (pruning): a árbore crece ata acadar o nº máximo de nodos, e logo suprímense pares de nodos terminais se así diminúe a entropía
- Sitúa os nodos terminais en varios niveis: árbores desequilibradas
- Nodos con tódolos fillos terminals pódense substituír unha subárbore por un nodo terminal



https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/

- Non está garantido que se minimice a entropía porque cada nodo usa unha soa entrada
- O podado non garante que se acade a árbore mínima

Árbores de regresión (I)

- En problemas de regresión, a entrada j a dividir e o umbral U da división selecciónanse de modo que se minimice a suma de erros cadráticos (SSE)
- Sexan $A_1(i,k)=\{l : x_{li}<k\}$ e $A_2(i,k)=\{l : x_{li} \geq k\}$: selecciónase (j,U) como:

$$(j,U)= \underset{1\leq i\leq n;\, k\in \boldsymbol{v}_i}{argmin} \left\{ \sum_{m\in A_1} (y_m - z_1)^2 + \sum_{m\in A_2} (y_m - z_2)^2 \right\}$$

$$z_p(i,k)= \frac{1}{N_p} \sum_{q\in A_p(i,k)} y_q, \; p=1,2$$

$\mathbf{v}_i$: vector cos valores de x_i

Predicción do nodo: media das saídas asociadas aos patróns deste nodo

N_p é o nº de elementos de $A_p(i,k)$

- Cando se crea un nodo terminal r, a súa saída y_r é o valor medio das saídas desexadas para os patróns de entrenamento $\{\mathbf{x}_{s(i)}, y_{s(i)}\}$, con $i=1..M_r$ que chegan a este nodo r:

$$y_r = \frac{1}{M_r} \sum_{i=1}^{M_r} y_{s(i)}$$

Árbores de regresión (II)

- A saída da árbore para un patrón de teste **x** é:

$$z(\boldsymbol{x}) = \sum_{i=1}^{R} y_i I_i(\boldsymbol{x})$$

- $R_1..R_R$ son as rexións nas que a árbore divide ao espazo de entrada, e R é o nº de rexións
- $I_i(\mathbf{x})=1$ se $\mathbf{x} \in R_i$ e $I_i(\mathbf{x})=0$ noutro caso
- Na práctica, o patrón **x** preséntase ao nodo raíz r e selecciónase o nodo fillo ao cal ir comparando a entrada x_k e o umbral U de r
- Se $x_k < U$ (resp. $x_k \geq U$), repítese o proceso con r_e (resp. r_d), fillo esquerdo (resp. dereito)
- O proceso repítese ata que se acada un nodo r terminal
- A saída $y(\mathbf{x})$ é y_r, a media das saídas desexadas para os patróns de entrenamento asociados ao nodo terminal r

Vantaxes e inconvintes

- ✔ Modelos de **caixa branca**: dan unha explicación do coñecemento aprendido (regras), e explican a saída predita
- ✔ Valen para entradas continuas e discretas: non requiren variábeis indicadoras (dummy)
- ✔ Realizan unha selección de entradas porque descartan as entradas non importantes, que non se dividen
- ✔ Son robustas con datos colineares (entradas dependentes)
- ✔ Funcionan ben con datos grandes
- ✗ Pouco estables: pequenos cambios no conxunto de entrenamento provocan grandes cambios na árbore
- ✗ Baseados en heurísticas: poden non acadar elevada precisión
- ✗ Poden xerar árbores moi grandes de difícil comprensión
- ✗ Isto pode provocar **sobreaprendizaxe**

Implementacións

- En Octave facemos unha implementación propia. En Matlab, require a Statistical and Machine Learning Toolbox

	Matlab	Python	R	Weka-Java
Árbore de clasificación	fitctree	tree/DecisionTreeClassifier	rpart/ rpart	weka.classifiers.trees.J48
Árbore de regresión	fitrtree	tree/DecisionTreeRegressor		weka.classifiers.trees.J48